Programmazione Orientata agli Oggetti in Linguaggio Java

Programmazione Grafica: Thread

versione 1.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Introduzione
- La Regola del Thread Singolo
- Esecuzione di Azioni Lunghe

Introduzione

- Relazione tra thread e Swing
  - ⇨molto forte, dal momento che i thread sono uno strumento centrale per la programmazione grafica
  - ⇨per la segnalazione degli eventi alla macchina virtuale
  - ⇨per la gestione degli eventi nell'applicazione ed il disegno dei componenti

Introduzione

- Due aspetti centrali di questo rapporto
  - ⇨la regola del thread singolo di Swing
  - ⇨l'esecuzione di operazioni lunghe
- Da cosa discendono
  - ⇨dal fatto che Swing NON è un package sicuro rispetto ai thread
- Package sicuro ("thread safe")
  - ⇨i metodi sono sincronizzati in modo da garantire la consistenza

Introduzione

- Il perchè di questa scelta
  - ⇨AWT è un package sicuro rispetto ai thread
  - ⇨Swing no, fondamentalmente per due ragioni
  - ⇨I ragione: prestazioni; sincronizzare rallenta le operazioni
  - ⇨II ragione: complessità; per evitare gli stalli, programmare con le librerie sicure è difficile
- Si tratta di un fenomeno già visto
  - ⇨java.util.Vector e Hashtable

Introduzione

- Come stanno davvero le cose
  - ⇨Swing non è quasi mai sincronizzato
  - ⇨ma alcuni metodi lo sono
  - ⇨ma AWT è completamente sincronizzato
  - ⇨e i componenti Swing ereditano dai componenti AWT
  - ⇨quindi la realtà è che il modello di esecuzione è misto (parzialmente sincronizzato)

La Regola del Thread Singolo

- Per risolvere il problema
  - ⇨Swing si basa su una regola
  - ⇨tutte le operazioni di disegno dei componenti devono avvenire in un unico thread
- Il Thread di Gestione degli Eventi
  - ⇨“Event Dispatching Thread” o “AWT-thread”
  - ⇨il thread di sistema in cui vengono avviati tutti i gestori di eventi (listener)

La Regola del Thread Singolo

- Il thread AWT
  - ⇨viene automaticamente avviato alla prima chiamata del metodo paint() o repaint()
  - ⇨questi metodi sono scritti in modo da essere sempre eseguiti nel thread AWT
- Da quel momento
  - ⇨il thread AWT resta in attesa (await()) che nella coda degli eventi siano inseriti eventi
  - ⇨li preleva e li notifica agli opportuni listener

La Regola del Thread Singolo

- La regola del thread singolo di Swing
  - ⇨per evitare corse e stalli, tutti i metodi di disegno dei componenti devono essere esegiuti nell'AWT-thread
- Non rispettare questa regola
  - ⇨è abbastanza semplice: basta chiamare metodi di disegno dei componenti dal main thread

La Regola del Thread Singolo

>> varie\framestallo\FrameStallo.java
>> dump dei thread

- Un esempio
  - ⇨un frame sviluppato come singleton che contiene al suo interno un JDesktopPane e un JInternalFrame
  - ⇨eseguito, conduce sistematicamente allo stallo
  - ⇨a causa della competizione tra il main thread e l'AWT-thread per il disegno dei componenti

```
public class FrameStallo extends javax.swing.JFrame {
   private static FrameStallo singleton;
   static {
     singleton = new FrameStallo();
    }
   private FrameStallo() {
      DesktopPane desktop = new DesktopPane();  // spostando queste istruzioni in basso
      getContentPane().add(desktop);                    // lo stallo può essere evitato
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setSize(300, 300);
      setVisible(true);
      setTitle("Frame Stallo");
      desktop.add(new javax.swing.JInternalFrame());
   }
   private class DesktopPane extends javax.swing.JDesktopPane  {
      protected void paintComponent(java.awt.Graphics g) {
         super.paintComponent(g);
         System.out.println("In attesa di esecuzione del metodo statico");
         getInstance();
      }
   }
   private static FrameStallo getInstance() {
      System.out.println("Questo metodo non verra' eseguito");
      return singleton;
   }
   public static void main(String[] args) {}
}
```


La Regola del Thread Singolo

- Per risolvere il problema
  - ⇨è necessario utilizzare i metodi della coda degli eventi
  - ⇨il metodo invokeLater() consente di inserire un evento nella coda degli eventi
  - ⇨il gestore dell'evento è il metodo run() di un oggetto di tipo runnable
  - ⇨come tutti gli altri gestori, il metodo run() viene eseguito nell'AWT

La Regola del Thread Singolo

```
public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new Principale().setVisible(true);
         }
      });
}
```

NOTA: il metodo run() non viene eseguito in un proprio thread, ma in un thread già esistente (l'AWT-thread)

```
public static void main(String args[]) {
      javax.swing.SwingUtilities.invokeLater(new Runnable() {
         public void run() {
            new Principale().setVisible(true);
         }
      });
}
```

in alternativa è possibile utilizzare il metodo invokeLater() della classe SwingUtilities

La Regola del Thread Singolo

- Nel caso del nostro FrameStallo
  - ⇨una possibile soluzione è costruire il frame nel thread AWT

```
public class FrameStallo extends javax.swing.JFrame {
   private static FrameStallo singleton;
   static {
      javax.swing.SwingUtilities.invokeLater(new Runnable() {
         public void run() {
            singleton = new FrameStallo();
         }
      });
   }
```

La Regola del Thread Singolo

- Nota
  - ⇨i metodi pericolosi sono i metodi di disegno
  - ⇨non i metodi che creano componenti prima di disegnarli
  - ⇨quindi è perfettamente sicuro inizializzare un frame o un pannello nel main thread prima di disegnarlo
  - ⇨bisogna poi disegnarlo nel thread AWT

Esecuzione di Azioni Lunghe

- Un problema molto serio
  - ⇨che discende dalla regola del thread singolo
  - ⇨il codice di disegno, la segnalazione degli eventi e la gestione degli eventi avviene in un unico thread
  - ⇨se un listener effettua un'operazione molto lunga, blocca il thread AWT e l'applicazione resta "congelata"
  - ⇨ovvero non risponde più ai gesti dell'utente

Esecuzione di Azioni Lunghe

>> mediapesataswing\AzioneExcelNoThread

- Un esempio: la media pesata
  - ⇨il comando “Apri con Excel”, che apre il documento creato con Microsoft Excel
- Per farlo
  - ⇨viene lanciato un nuovo processo utilizzando le classi Runtime e Process
  - ⇨per rendere lunga l’esecuzione del listener, viene usato il metodo waitFor()
  - ⇨l’applicazione resta completamente bloccata

Esecuzione di Azioni Lunghe

- La soluzione corretta
  - ⇨eseguire le azioni “lunghe” in thread separati in modo da evitare di appesantire il thread AWT
- Ma attenzione...
  - ⇨nel thread separato non è possibile eseguire operazioni di disegno, per evitare di violare la regola del thread singolo

Esecuzione di Azioni Lunghe

- I caso tipico
  - ⇨nell'azione viene avviato un thread separato creando un runnable il cui metodo run() contiene il codice dell'azione "lunga"
  - ⇨durante l'azione lunga è necessario ridisegnare alcuni componenti (es: barra di avanzamento)
  - ⇨in questo caso dal nuovo thread è possibile eseguire codice nel thread AWT utilizzando EventQueue.invokeLater()

Esecuzione di Azioni Lunghe

- II caso tipico
  - ⇨è necessario eseguire un'azione lunga, e al termine produrre uno schermo che contiene il risultato
  - ⇨questo è un problema più complesso, perchè richiede la sincronizzazione tra i due thread
- La soluzione
  - ⇨la classe SwingWorker

Esecuzione di Azioni Lunghe

- SwingWorker
  - una classe non ufficiale, di “pubblico dominio”
  - è possibile estenderla ridefinendo due metodi
  - construct(): operazioni da eseguire in un thread separato
  - finished(): operazioni da eseguire al termine nel thread AWT

Esecuzione di Azioni Lunghe

>> mediapesataswing\AzioneExcel

- Un esempio
  - AzioneExcel
  - risolve il problema del blocco dell’interfaccia utilizzando SwingWorker
- ping
  - ha una sua classe PingThreadWorker
  - basata su SwingWorker
  - nel package it.unibas.ping.contrib

Riassumendo

- Introduzione
- La Regola del Thread Singolo
- Esecuzione di Azioni Lunghe

Termini della Licenza